



## IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:  
Eitan Marcus et. al

Application No.: 10/040,940

Filed: 01/09/02

Art Unit: 2863

For: Adaptive Test Generation

Examiner: Xiuqin Sun

Mail Stop AF  
Commissioner for Patents  
PO Box 1450  
Alexandria, VA 22313-1450

## DECLARATION UNDER 37 CFR 1.131

Sir:

1) We, the undersigned, Allon Adir, Roy Emek, and Eitan Marcus hereby declare as follows:

2) We are the Applicants in the patent application identified above, and are co-inventors of the subject matter described and claimed in claims 54-71 therein.

3) Subsequent to January 1, 1996 and prior to August 24, 2001, we reduced our invention to practice, as described and claimed in the subject application, in Israel, a WTO country. We operated a verification system, which was IBM's Genesys-Pro system. This system included a generator of test programs for veri-

fying a design for compliance with its design specification, as described and claimed in the subject Application.

4) Operation of the verification system is corroborated by Michael Vinov, whose Declaration is attached hereto as Exhibit 1, and which provides further details of the verification system's operation and other Exhibits referred to herein. Michael Vinov has first hand knowledge of the claimed subject matter.

5) Tables 1 - 2 show the correspondence between the elements of apparatus claims in the present patent application and the material in the Exhibits.

Table 1

Claim 66	Disclosure
A test program generator, comprising:	Exhibit 5, Sec. 1.1. Fig. 1 illustrates the architecture of the Genesys-Pro test program generation system
a test program generation engine;	Exhibit 5, Sec. 1.1. Fig. 1 shows a test program generator.
a design specification of a target, wherein said design specification comprises a knowledge base, wherein said test program generation engine is coupled to said design specification;	Exhibit 5, Sec. 1.1. An architectural model containing testing knowledge is indicated in Fig. 1. Coupling of the test program generation engine to the model is illustrated.
an architectural simulator of said target coupled to said	Exhibit 5, Sec. 1.1. An architectural simulator is ex-

test program generation engine;	explicitly shown in Fig. 1
said test program generation engine and said architectural simulator being cooperative to perform a method of test program generation for a system-under-test, comprising the steps of:	Exhibit 5, Sec. 1.1. All the components of the Genesys-Pro test generation system are indicated as being interacting.
using a primary input stream to generate a sequence of test program instructions for said system-under-test;	Exhibit 2. The file table_walk.def describes the primary input stream to the generator
loading a set of events, each event in said set having a triggering condition and a predetermined alternate input stream, said primary input stream and said alternate input stream comprising sequences of partially specified program instructions for said system-under-test;	Exhibit 2. The files, default.ih and handlers.ih, contain alternative input streams that contain a sequence of partially specified instructions for the IBM Power4 processor. Both these files are event files, and the file default.ih is pointed-to by the primary input stream - table_walk.def.
recognizing that said triggering condition of one of said events is satisfied;	Exhibit 5. The ability of the Genesys-Pro test generation system to recognize triggering conditions of events is shown generally in Sec. 4.6.2 and Sec 5.3.  Exhibit 3. Exceptions re-

	flecting recognition of triggering conditions are shown in the output file table_walk.tst at page E3-3.
responsively to said step of recognizing, selecting one of said primary input stream and said alternate input stream as a continuation input stream; and	Exhibit 3. At page E3-3 of the listing, which is shown in bold-face type for convenience, is a section of the program that was generated according to an alternative input steam - taken from the file handlers.ih (Exhibit 2), which was selected as the continuation input stream. This section is marked by "* E Exception" in its beginning and end.
generating subsequent test program instructions using said continuation input stream.	Exhibit 3. The listing at page E3-3 show instructions that are generated using the continuation input stream.

Table 2

Claim 67	Disclosure
The test program generator according to claim 66, wherein a portion of said set of events are included in said primary input stream.	Exhibit 5; Sec. 4.6.2 explains that an event can be included in a def file, which can be a primary input stream.

Claim 68	Disclosure
The test program generator according to claim 66, wherein said set of events is loaded from an event file.	Exhibit 2. An event file is specified in last line of the file default.ih.
Claim 69	Disclosure
The test program generator according to claim 66 wherein an occurrence of said triggering condition occurs nonpredeterminedly in said primary input stream.	Exhibit 5. Sec. 4.6.2 Events in the input stream may be introduced in a def file. Random inclusion is allowed.
Claim 70	Disclosure
The test program generator according to claim 66, wherein said events have priority values, and are processed in order of said priority values.	Exhibit 5. Sec. 4.6.2. Genesys-Pro iterates over its list of events to see if their triggering condition is now TRUE.

6) These tables demonstrate that we conceived the entire invention, as recited in claims 66 - 70, prior to August 24, 2001.

5 Based on the similarity of subject matter between claims 66 - 71 and claims 54 - 65 and 71 - 75, it can similarly be demonstrated that we actually reduced to practice the entire invention recited in claims 54 - 65 and 71 - 75.

7) All dates deleted from the text and exhibits hereof are prior to August 24, 2001.

8) We hereby declare that all statements made herein of our own knowledge are true and that all statements made on information and conjecture are thought to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application of any patent issued thereon.

21.11.04  
Allon Adir, Citizen of Israel,  
70 Allonim Street, Tivon, Israel

Date: 21-11-2004

22.11.04  
Roy Emek, Citizen of Israel,  
32 Oranim Street, Kfar  
Shmaryahu, Israel

Date: 22-11-04

Ad. Marcus  
Eitan Marcus, Citizen of Israel,  
5 Eder Street, Haifa, Israel

Date: 22.11.2004

**Exhibit 1**

**DECLARATION OF MICHAEL VINOV**

5 Sir:

I, the undersigned, Michael Vinov, hereby declare as follows:

10 1) I am an employee of International Business Machines Corporation, the assignee hereof, and work at IBM Haifa Labs, Haifa, Israel. My present responsibilities include supervision of aspects of the research and development activities of the Applicants of the Application hereof. At all times material  
15 hereto, I was a member of a project development team relating to the invention disclosed and claimed in the Application. I am familiar with the subject matter of the Application.

20 2) Subsequent to January 1, 1996 and prior to August 24, 2001, I observed the operation of a verification system in Israel, a WTO country, as further detailed in the accompanying Declaration of the Inventors pursuant to 37 C.F.R. 1.131. The verification system included a generator of test programs for verifying a design for compliance with its design specification,  
25 as described and claimed in the subject Application.

30 3) Some of the inputs to the system during the observed operation were files named table\_walk.def, default.ih, and handlers.ih, which are attached hereto as Exhibit 2. The first file, table\_walk.def, describes the primary input stream to the

generator, while the other two files, default.ih and handlers.ih, contain alternative input streams to be used for various events. The target design-under-test was, in this case, IBM's Power4 processor.

5

4) The output of the system was captured in a file table\_walk.tst, attached hereto as Exhibit 3. In a portion of the listing, at page E3-3, which is shown in bold-face type for convenience, is a section of the program that was generated according to an alternative input steam - taken from handlers.ih. This section is marked by "\* E Exception" in its beginning and end

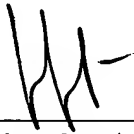
5) A display of the subdirectory containing the files table\_walk.def and table\_walk.tst indicating their dates of last modification is included herewith as Exhibit 4. In addition, the first line of the file table\_walk.tst itself indicates its generation time.

6) Exhibit 5 contains selected portions of the Genesys-Pro documentation, dated [REDACTED], which discusses the architecture of Genesys-Pro (Sec. 1.1), events generally (Sec. 4.6.2), and the usage of events in the context of interrupt handlers (Sec. 5.3). This documentation was applicable at the time operation of the system was observed, and it shows that the Genesys-Pro system to be capable of operation according to the claimed invention.

7) Successful operation of the test generator in accordance with the claims hereof was indicated by the test file in Exhibit 3.



8) I hereby declare that all statements made herein of my  
own knowledge are true and that all statements made on informa-  
tion and conjecture are thought to be true; and further that  
5 these statements were made with the knowledge that willful false  
statements and the like so made are punishable by fine or im-  
prisonment, or both, under Section 1001 of Title 18 of the  
United States Code and that such willful false statements may  
jeopardize the validity of the application of any patent issued  
10 thereon.



---

Michael Vinov  
Citizen of Israel

Address:  
8a Watson St.,  
Haifa, Israel

Date:

---

22/11/04

## Exhibit 2

## table\_walk.def

```

5
<?xml version="1.0"?>
<!DOCTYPE GPRODEF SYSTEM "gprodef.dtd" >
<GPRODEF testFile="table_walk.tst" version="1.2.4">
  <includeFile fileName="default.ih"/>
10  <TEST initialState="Real_Mode">
    <CONCURRENT>
      <PROCESS id="0,0">
        <SEQUENCE>
          <varDecl type="bitstream" name="address"/>
15  <ASSIGN value="0x3000" name="address"/>
          <REPEAT condition="$address SUBSETOF {[0x3000,0x4000]}">
            <SEQUENCE>
              <l name="ld" mandatory="FALSE">
                <OP name="base_displacement">
20  <PROPERTY value="$address" name="address"/>
                </OP>
              </l>
              <ASSIGN value="$address + 0x100" name="address"/>
            </SEQUENCE>
25  </REPEAT>
          </SEQUENCE>
        </PROCESS>
      </CONCURRENT>
    </TEST>
30 </GPRODEF>

```

**default.ih**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE EVENT_FILE SYSTEM "gprodef.dtd" >
5  <EVENT_FILE version="1.2.6">
    <varDecl type="bool" initValue="FALSE" name="AfterEpilogue"/>
    <varDecl type="string" initValue="&quot;GP&quot;" name="Design_Name"/>
    <includeFile fileName="default.mac"/>

10  <!-- Important Note: Handler events must be listed before reloading events
    so that they can be first to fire -->

    <includeFile fileName="handlers.ih"/>
    <includeFile fileName="reloading.ih"/>
15  <includeFile fileName="redefinition.ih"/>

</EVENT_FILE>
```

**handlers.ih**

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE EVENT_FILE SYSTEM "gprodef.dtd" >
5  <EVENT_FILE version="1.2.2">

    <reserveResource address="0x000000000000CE0"
        class="MEMORY" resource="MEMORY" numOfUnits="8"/>
    -->
10  <varDecl type="string" initValue="0x02" name="num_procs" />

    <EVENT reservedRangeSize="16" name="MACHINE_CHECK" condition="(Exception() &#38;&#38;
        ($IAR == 0x000000000000200))"
        reservedRangeStart="000000000000200">
15  <SEQUENCE reentrant="TRUE" regenerate="FALSE">
    <directive value="10" name="pure"/>
    <PRINT expr="&#34;%% Machine Check&#34;"/>
    <VERBATIM>
    <instrCard mnemonic="&#34;mfsrr0 G28&#34;"/>
20  <instrCard mnemonic="&#34;addi G28,G28,0x4&#34;"/>
    <instrCard mnemonic="&#34;mtsrr0 G28&#34;"/>
    </VERBATIM>
    <l name="rfid">
    <directive value="rfi_bnt_macro" name="bnt_macro_call"/>
25  </l>
    </SEQUENCE>
    </EVENT>

30  <EVENT reservedRangeSize="24" name="DSI" condition="(Exception() &#38;&#38;
        ($IAR == 0x000000000000300))"
        reservedRangeStart="000000000000300">
    <SEQUENCE reentrant="TRUE" regenerate="FALSE">
    <directive value="10" name="pure"/>
35  <PRINT expr="&#34;%% DSI&#34;"/>
    <VERBATIM>
    <instrCard mnemonic="&#34;mfsrr0 G28&#34;"/>
    <instrCard mnemonic="&#34;addi G28,G28,0x4&#34;"/>
    <instrCard mnemonic="&#34;mtsrr0 G28&#34;"/>
40  <instrCard mnemonic="&#34;mtdar G28&#34;"/>
    <instrCard mnemonic="&#34;mtdsir G28&#34;"/>
    </VERBATIM>
    <l name="rfid">
    <directive value="rfi_bnt_macro" name="bnt_macro_call"/>
45  </l>
    </SEQUENCE>
    </EVENT>

50  <EVENT reservedRangeSize="24" name="DATA_SEGMENT" condition="(Exception() &#38;&#38;
        ($IAR == 0x000000000000380))"
        reservedRangeStart="000000000000380">
    <SEQUENCE reentrant="TRUE" regenerate="FALSE">
    <directive value="10" name="pure"/>

```

```

<PRINT expr="&#34;%% Data Segment&#34;"/>
<VERBATIM>
<instrCard mnemonic="&#34;mfsrr0 G28&#34;"/>
<instrCard mnemonic="&#34;addi G28,G28,0x4&#34;"/>
5 <instrCard mnemonic="&#34;mtsrr0 G28&#34;"/>
<instrCard mnemonic="&#34;mtdar G28&#34;"/>
<instrCard mnemonic="&#34;mtdsir G28&#34;"/>
</VERBATIM>
<l name="rfid">
10 <directive value="rfi_bnt_macro" name="bnt_macro_call"/>
</l>
</SEQUENCE>
</EVENT>

15 <EVENT reservedRangeSize="40" name="ISI" condition="(Exception() &#38;&#38;
($IAR == 0x00000000000000400) )"
reservedRangeStart="00000000000000400">
<SEQUENCE reentrant="TRUE" regenerate="FALSE">
<directive value="10" name="pure"/>
20 <PRINT expr="&#34;%% ISI&#34;"/>
<VERBATIM>
<initResource address="0" class="REGISTER" resource="General_Purpose_Register"/>
<instrCard mnemonic="&#34;addis G27,G0,0x1010&#34;"/>
<instrCard mnemonic="&#34;mfsrr0 G28&#34;"/>
25 <instrCard mnemonic="&#34;add G28,G28,G27&#34;"/>
<instrCard mnemonic="&#34;addi G28,G28,0x40&#34;"/>
<instrCard mnemonic="&#34;rdicl G28,G28,0x00,0x00&#34;"/>
<instrCard mnemonic="&#34;mtsrr0 G28&#34;"/>
<instrCard mnemonic="&#34;mfsrr1 G28&#34;"/>
30 <instrCard mnemonic="&#34;ori G28,G28,0x0020&#34;"/>
<instrCard mnemonic="&#34;mtsrr1 G28&#34;"/>
</VERBATIM>
<l name="rfid">
<directive value="rfi_bnt_macro" name="bnt_macro_call"/>
35 </l>
</SEQUENCE>
</EVENT>

<EVENT reservedRangeSize="40" name="INSTRUCTION_SEGMENT" condition="(Exception()
40 &#38;&#38;
($IAR == 0x00000000000000480) )"
reservedRangeStart="00000000000000480">
<SEQUENCE reentrant="TRUE" regenerate="FALSE">
<directive value="10" name="pure"/>
45 <PRINT expr="&#34;%% Instruction Segment&#34;"/>
<VERBATIM>
<initResource address="0" class="REGISTER" resource="General_Purpose_Register"/>
<instrCard mnemonic="&#34;addis G27,G0,0x1010&#34;"/>
<instrCard mnemonic="&#34;mfsrr0 G28&#34;"/>
50 <instrCard mnemonic="&#34;add G28,G28,G27&#34;"/>
<instrCard mnemonic="&#34;addi G28,G28,0x40&#34;"/>
<instrCard mnemonic="&#34;rdicl G28,G28,0x00,0x00&#34;"/>
<instrCard mnemonic="&#34;mtsrr0 G28&#34;"/>
<instrCard mnemonic="&#34;mfsrr1 G28&#34;"/>

```

```

    <instrCard mnemonic="&#34;ori G28,G28,0x0020&#34;"/>
    <instrCard mnemonic="&#34;mtsrr1 G28&#34;"/>
  </VERBATIM>
  <l name="rfid">
5    <directive value="rfi_bnt_macro" name="bnt_macro_call"/>
    </l>
  </SEQUENCE>
</EVENT>

10  <EVENT reservedRangeSize="16" name="ALIGNMENT" condition="(Exception() &#38;&#38;
      ($IAR == 0x0000000000000600) )"
      reservedRangeStart="0000000000000600">
    <SEQUENCE reentrant="TRUE" regenerate="FALSE">
      <directive value="10" name="pure"/>
      <PRINT expr="&#34;%%&#34; Alignment&#34;"/>
      <VERBATIM>
        <instrCard mnemonic="&#34;mfsrr0 G28&#34;"/>
        <instrCard mnemonic="&#34;addi G28,G28,0x4&#34;"/>
        <instrCard mnemonic="&#34;mtsrr0 G28&#34;"/>
      </VERBATIM>
      <l name="rfid">
        <directive value="rfi_bnt_macro" name="bnt_macro_call"/>
        </l>
      </SEQUENCE>
25  </EVENT>

    <EVENT reservedRangeSize="160" name="PROGRAM_INTERRUPT" condition="(Exception()
&#38;&#38;
      ($IAR == 0x0000000000000700) )"
      reservedRangeStart="0000000000000700">
30    <SEQUENCE reentrant="TRUE" regenerate="FALSE">
      <directive value="10" name="pure"/>
      <PRINT expr="&#34;%%&#34; 700&#34;"/>
      <VERBATIM>
        <initResource address="28" class="REGISTER" resource="General_Purpose_Register"/>
        <instrCard mnemonic="&#34;cmpi 0,0,G28,0x3025&#34;"/>
        <instrCard mnemonic="&#34;bc 0x04,0x02,0x0018&#34;"/>
        <instrCard mnemonic="&#34;slbmte G5,G6&#34;"/>
        <instrCard mnemonic="&#34;mfsrr0 G28&#34;"/>
        <instrCard mnemonic="&#34;addi G28,G28,0x4&#34;"/>
        <instrCard mnemonic="&#34;mtsrr0 G28&#34;"/>
        <instrCard mnemonic="&#34;rfid&#34;"/>
        <instrCard mnemonic="&#34;mfsrr1 G28&#34;"/>
        <instrCard mnemonic="&#34;andis. G29,G28,0x0010&#34;"/>
        <instrCard mnemonic="&#34;bc 0x04,0x02,0x0014&#34;"/>
        <instrCard mnemonic="&#34;mfsrr0 G28&#34;"/>
        <instrCard mnemonic="&#34;addi G28,G28,0x4&#34;"/>
        <instrCard mnemonic="&#34;mtsrr0 G28&#34;"/>
        <instrCard mnemonic="&#34;rfid&#34;"/>
45    </VERBATIM>
    </SEQUENCE>
50  -->
    <instrCard mnemonic="&#34;andis. G28,G28,0x0001&#34;"/>
    <instrCard mnemonic="&#34;bc 0x04,0x02,0x0010&#34;"/>
    <instrCard mnemonic="&#34;mfsrr0 G28&#34;"/>
    <instrCard mnemonic="&#34;addi G28,G28,0x4&#34;"/>

```

```

<instrCard mnemonic="#34;mtsr0 G28&#34;"/>
<instrCard mnemonic="#34;mfmrsr G28&#34;"/>
<instrCard mnemonic="#34;ori G28,G28,0x2000&#34;"/>
<instrCard mnemonic="#34;mtmsrd G28,0x0&#34;"/>
5 <instrCard mnemonic="#34;isync&#34;"/>
<!-- Clear FPSCR exception bits -->
<instrCard mnemonic="#34;mtfsfi 0x00,0x00&#34;"/>
<instrCard mnemonic="#34;mtfsfi 0x01,0x00&#34;"/>
<instrCard mnemonic="#34;mtfsfi 0x02,0x00&#34;"/>
10 <instrCard mnemonic="#34;mtfsfi 0x03,0x00&#34;"/>
<instrCard mnemonic="#34;mtfsfi 0x05,0x00&#34;"/>
<instrCard mnemonic="#34;rfd&#34;"/>
</VERBATIM>
</SEQUENCE>
15 </EVENT>

<EVENT reservedRangeSize="16" name="FP_UNAVAILABLE" condition="(Exception() &#38;&#38;
($IAR == 0x0000000000000800) )"
reservedRangeStart="0000000000000800">
20 <SEQUENCE reentrant="TRUE" regenerate="FALSE">
<directive value="10" name="pure"/>
<PRINT expr="#34;%% Floating Point not available&#34;"/>
<VERBATIM>
<instrCard mnemonic="#34;mfsrr0 G28&#34;"/>
25 <instrCard mnemonic="#34;addi G28,G28,0x4&#34;"/>
<instrCard mnemonic="#34;mtsr0 G28&#34;"/>
</VERBATIM>
<l name="rfd">
<directive value="rfi_bnt_macro" name="bnt_macro_call"/>
30 </l>
</SEQUENCE>
</EVENT>

<EVENT reservedRangeSize="4" name="DECREMENTER_INTERRUPT" condition="(Exception()
&#38;&#38;
35 ($IAR == 0x0000000000000900) )"
reservedRangeStart="0000000000000900">
<SEQUENCE reentrant="TRUE" regenerate="FALSE">
<directive value="10" name="pure"/>
40 <PRINT expr="#34;%% Decrementer Interrupt&#34;"/>
<l name="rfd">
<directive value="rfi_bnt_macro" name="bnt_macro_call"/>
</l>
</SEQUENCE>
45 </EVENT>

<EVENT reservedRangeSize="96" name="SYSTEM_CALL" condition="(Exception() &#38;&#38;
50 ($IAR == 0x0000000000000C00) )"
reservedRangeStart="0000000000000C00">
<SEQUENCE reentrant="TRUE" regenerate="FALSE">
<directive value="10" name="pure"/>
<directive value="TRUE" name="unguarded"/>

```

```

<ASSIGN name="num_procs" value="String(Choose(NumOfProcesses()))"/>
<SELECT>
<SEQUENCE precondition="NumOfProcesses() != 1">
  <PRINT expr="&#34;%% System Call MP_SYNCHRONIZATION&#34;"/>
5  <VERBATIM>
    <!-- The doubleword at 0x0CE0 stores two semaphores (one in bits 0:15, the other -->
    <!-- in bits 16:31) and a shift amount in bits 58:63 indicating which semaphore -->
    <!-- is in use. The barriers alternate between semaphores so that barrier N -->
    <!-- does not prematurely modify the semaphore being polled in barrier N-1. -->
10  <initResource address="0x000000000000CE0" class="MEMORY" resource="MEMORY" num_of_u-
    nits="8" value="0x00000000000030"/>
    <!-- WIMG=3 semaphores + shift amount -->
    <initResource address="0" class="REGISTER" resource="General_Purpose_Register"/>
    <!-- force visibility to other processors -->
15  <instrCard mnemonic="&#34;sync&#34;"/>
    <instrCard mnemonic="&#34;addi G26,G0,0x0CE0&#34;"/>
    <instrCard mnemonic="&#34;ldarx G27,G0,G26&#34;"/>
    <instrCard mnemonic="&#34;srd G28,G27,G27&#34;"/>
    <instrCard mnemonic="&#34;andi. G28,G28,0xFFFF&#34;"/>
20  <instrCard mnemonic="&#34;cmpi 0x00,0x00,G28,&#34; # $num_procs "/>
    <instrCard mnemonic="&#34;bc 0x04,0x02,0010&#34;"/>
    <!-- semaphore still set from last barrier; switch to alternate barrier and clear it -->
    <instrCard mnemonic="&#34;sld G28,G28,G27&#34;"/>
    <instrCard mnemonic="&#34;rlwimi G28,G27,0x00,0x1A,0x1F&#34;"/>
25  <instrCard mnemonic="&#34;xori G27,G28,0x0010&#34;"/>
    <!-- increment semaphore and attempt to store -->
    <instrCard mnemonic="&#34;addi G28,G0,0x0001&#34;"/>
    <instrCard mnemonic="&#34;sld G28,G28,G27&#34;"/>
    <instrCard mnemonic="&#34;add G27,G27,G28&#34;"/>
30  <instrCard mnemonic="&#34;stdcx. G27,G0,G26&#34;"/>
    <instrCard mnemonic="&#34;bc 0x04,0x02,0xFFD0&#34;"/>
    <!-- poll semaphore until equal to number of processors -->
    <instrCard mnemonic="&#34;ldx G28,G0,G26&#34;"/>
    <instrCard mnemonic="&#34;srd G28,G28,G27&#34;"/>
35  <instrCard mnemonic="&#34;andi. G28,G28,0xFFFF&#34;"/>
    <instrCard mnemonic="&#34;cmpi 0x00,0x00,G28,&#34; # $num_procs "/>
    <instrCard mnemonic="&#34;nop&#34;"/>
    <!-- clobber CR0, G27, G28 to make predictable -->
    <instrCard mnemonic="&#34;addi G27,G0,0x0000&#34;"/>
40  <instrCard mnemonic="&#34;addi G28,G0,0x0000&#34;"/>
    <instrCard mnemonic="&#34;cmp 0x00,0x00,G27,G28&#34;"/>
    </VERBATIM>
    </SEQUENCE>
    <SEQUENCE precondition="NumOfProcesses() == 1">
45  <PRINT expr="&#34;%% System Call UNIPROCESSOR (no synch)&#34;"/>
    </SEQUENCE>
    </SELECT>
    <l name="rfid">
    <directive value="rfi_bnt_macro" name="bnt_macro_call"/>
50  </l>
    </SEQUENCE>
  </EVENT>

<EVENT reservedRangeSize="4" name="TRACE_INTERRUPT" condition="(Exception() &#38;&#38;

```



```

                                ($IAR == 0x000000000000D00) )"
                                reservedRangeStart="000000000000D00">
5    <SEQUENCE reentrant="TRUE" regenerate="FALSE">
    <directive value="10" name="pure"/>
    <PRINT expr="&#34;%% Trace Interrupt&#34;"/>
    <l name="rfid">
    <directive value="rfi_bnt_macro" name="bnt_macro_call"/>
    </l>
    </SEQUENCE>
10   </EVENT>

    <EVENT reservedRangeSize="16" name="INSTRUCTION_ADDRESS_BREAKPOINT" condi-
    tion="(Exception() &#38;&#38;
                                ($IAR == 0x0000000000001300) )"
                                reservedRangeStart="0000000000001300">
15   <SEQUENCE reentrant="TRUE" regenerate="FALSE">
    <directive value="10" name="pure"/>
    <PRINT expr="&#34;%% Instruction Address Breakpoint&#34;"/>
    <VERBATIM>
20   <instrCard mnemonic="&#34;mfsrr0 G28&#34;"/>
    <instrCard mnemonic="&#34;addi G28,G28,0x4&#34;"/>
    <instrCard mnemonic="&#34;mtsrr0 G28&#34;"/>
    </VERBATIM>
    <l name="rfid">
25   <directive value="rfi_bnt_macro" name="bnt_macro_call"/>
    </l>
    </SEQUENCE>
    </EVENT>

30   <EVENT reservedRangeSize="16" name="INSTRUMENTATION_INTERRUPT" condition="(Exception()
    &#38;&#38;
                                ($IAR == 0x0000000000002000) )"
                                reservedRangeStart="0000000000002000">
    <SEQUENCE reentrant="TRUE" regenerate="FALSE">
35   <directive value="10" name="pure"/>
    <PRINT expr="&#34;%% Instrumentation Interrupt&#34;"/>
    <l name="rfid">
    <directive value="rfi_bnt_macro" name="bnt_macro_call"/>
    </l>
40   </SEQUENCE>
    </EVENT>

</EVENT_FILE>

```

## Exhibit 3

## table\_walk.tst

\* Genesys Pro Test File

5 \* Produced on: [REDACTED]

\*

TEST 1001

H Seed: 2751470673 System: PowerPC Version: Format:

10 INITIALIZATIONS: DATA MEMORY (MEMORY)

D 0000000000003000 BB6468800821D2BA

D 0000000000003100 BE65C5F7DA43DFBE

D 0000000000003200 8EE3A86B37E2C5E9

D 0000000000003300 471C748F37EBFD8A

15 D 0000000000003400 0B878BECE9CB81EB

D 0000000000003500 2471657C6826B413

D 0000000000003600 1802F99900C52857

D 0000000000003700 2B36EEC45420BE12

D 0000000000003800 5CC77810154A4BCF

20 D 0000000000003900 0BC1F2A5DF45CEF7

D 0000000000003A00 96CFED829B251CF8

D 0000000000003B00 9608082B9C2F3D9E

D 0000000000003C00 D3B5720D5D85E7D0

D 0000000000003D00 33945300C89F9CA8

25 D 0000000000003E00 DF16182FB73809E7

D 0000000000003F00 B4281C2F814ECB1D

D 0000000000004000 4B8BB7DB5A819822

TAG 000000000000300 1

TAG 000000000000310 0

30 TAG 0000000000003000 1

TAG 0000000000003100 0

TAG 0000000000003200 1

TAG 0000000000003300 0

TAG 0000000000003400 0

35 TAG 0000000000003500 0

TAG 0000000000003600 1

TAG 0000000000003700 0

TAG 0000000000003800 0

TAG 0000000000003900 0

40 TAG 0000000000003A00 1

TAG 0000000000003B00 1

TAG 0000000000003C00 1

TAG 0000000000003D00 1

TAG 0000000000003E00 1

45 TAG 0000000000003F00 1

TAG 0000000000004000 1

TAG 0000011260248D30 1

TAG 0000011260248D40 0

TAG 0000011260248D50 1

50 TAG 0000011260248D60 0

TAG 0000011260248D70 1

TAG 0000011260248D80 0  
 TAG 000001307EAF2A0 1  
 TAG 000001307EAF2B0 1  
 CLUSTER 0

5  
 PROCESS 0  
 INITIALIZATIONS: REGISTERS  
 R ACCR 0000000000000000  
 R CacheConfig 0000000100800080  
 10 R CR 9395820D  
 R CTRL 00000001  
 R CTR FD7E1CCAB56A2637  
 R DABR 743B63A181B84059  
 R DAR 74BC0B71525E05D0  
 15 R DSISR 8F69D49F  
 R XER 00001E6708E00010  
 R FPSCR 80037001  
 R G1 FCF926C0C54A62B4  
 R G2 B21B889EB883914F  
 20 R G3 FFFFFFFFFFBF80  
 R G4 41060254B2252F8D  
 R G5 00000000000009AEC  
 R G7 D0B8F8D86C8399F6  
 R G11 FFFFFFFFFFC9D0  
 25 R G13 B7EA0B805C310DB5  
 R G14 0000000000005E4C  
 R G15 00000000000014F0  
 R G16 027DAC7577921364  
 R G17 0000000000009A8C  
 30 R G18 8F367B0AD842622E  
 R G19 FFFFFFFFFFE14  
 R G20 8C67578FA883EF6C  
 R G21 188E004A3685C3E0  
 R G22 6262D4750F430364  
 35 R G23 0000000000009AE8  
 R G24 8592237E2C78C395  
 R G25 82D1F90D4629BFE3  
 R G27 0000000000002EAC  
 R G28 4608DDF468F31503  
 40 R G30 0000000000004F30  
 R G31 D227C57EB8511106  
 R HID0 0000000100000000  
 R HID1 F91C000000000000  
 R HID4 8B60000000000011  
 45 R HID5 8000000000000010  
 R IAR EF15111260248D38  
 R LR 000ADB9349BD5A65  
 R MSR D000000000002800  
 R SRR0 BCAE6B9DDF979D79  
 50 R SRR1 8DE950F6F38E8843  
 R MMCR0 0000000080000000  
 R PIR 00000000  
 R RESRV D85A4A6C5226E350  
 R SDAR BF7CE4C07B15444A

R SIAR        CCBBA3970BFEEB78  
 R SDR1        0000000000040000  
 R\_SIM        0000000000000000

```

5  PHASE 0  INSTRUCTIONS
   I 0000011260248D38 EB203000 * EA=EF15111260248D38 ld G25,0x3000(G0)
   I 0000011260248D3C EAD344EC * EA=EF15111260248D3C ld G22,0x44EC(G19)
   * E Exception
   I 0000000000000300 7F9A02A6 * EA=0000000000000300 mfsrr0 G28
10  I 0000000000000304 3B9C0004 * EA=0000000000000304 addi G28,G28,0x4
   I 0000000000000308 7F9A03A6 * EA=0000000000000308 mtsrr0 G28
   I 000000000000030C 7F9303A6 * EA=000000000000030C mtdar G28
   I 0000000000000310 7F9203A6 * EA=0000000000000310 mtdsisr G28
   I 0000000000000314 4C000024 * EA=0000000000000314 rfid
15  I 0000011260248D40 E9AED3B4 * EA=EF15111260248D40 ld G13,0xD3B4(G14)
   I 0000011260248D44 E8437380 * EA=EF15111260248D44 ld G2,0x7380(G3)
   * E Exception
   * Reentering at 0000000000000300
   I 0000011260248D48 EA979918 * EA=EF15111260248D48 ld G20,0x9918(G23)
20  I 0000011260248D4C EA4F2010 * EA=EF15111260248D4C ld G18,0x2010(G15)
   I 0000011260248D50 EAC03600 * EA=EF15111260248D50 ld G22,0x3600(G0)
   I 0000011260248D54 EA003700 * EA=EF15111260248D54 ld G16,0x3700(G0)
   I 0000011260248D58 E83EE8D0 * EA=EF15111260248D58 ld G1,0xE8D0(G30)
   I 0000011260248D5C E8203900 * EA=EF15111260248D5C ld G1,0x3900(G0)
25  I 0000011260248D60 E8EB7030 * EA=EF15111260248D60 ld G7,0x7030(G11)
   * E Exception
   * Reentering at 0000000000000300
   I 0000011260248D64 EAB1A074 * EA=EF15111260248D64 ld G21,0xA074(G17)
   I 0000011260248D68 E8E5A114 * EA=EF15111260248D68 ld G7,0xA114(G5)
30  I 0000011260248D6C EB1B0E54 * EA=EF15111260248D6C ld G24,0x0E54(G27)
   I 0000011260248D70 E8803E00 * EA=EF15111260248D70 ld G4,0x3E00(G0)
   I 0000011260248D74 EBE03F00 * EA=EF15111260248D74 ld G31,0x3F00(G0)
   I 0000011260248D78 EAA04000 * EA=EF15111260248D78 ld G21,0x4000(G0)

35  EPILOGUE
   * Begin macro Epilogue_Sequence
   I 0000011260248D7C 7C0004AC * EA=EF15111260248D7C sync_NE
   I 0000011260248D80 7C0004AC * EA=EF15111260248D80 sync_NE
   I 0000011260248D84 7C0004AC * EA=EF15111260248D84 sync_NE
40  I 0000011260248D88 F000006E * EA=EF15111260248D88 notrace
   * End of macro Epilogue_Sequence
   *****

RESULTS: REGISTERS
R ACCR        0000000000000000
45  R CacheConfig 0000000100800080
   R CR        9395820D
   R CTRL      00000001
   R CTR       FD7E1CCAB56A2637
   R DABR      743B63A181B84059
50  R DAR       EF15111260248D64
   R DSISR     60248D64
   R XER       00001E6708E00010
   R FPSCR     80037001
   R G1        0BC1F2A5DF45CEF7

```

```

R G2      B21B889EB883914F
R G3      FFFFFFFFFFBF80
R G4      DF16182FB73809E7
R G5      0000000000009AEC
5  R G7      D3B5720D5D85E7D0
R G11     FFFFFFFFFFC9D0
R G13     8EE3A86B37E2C5E9
R G14     0000000000005E4C
R G15     00000000000014F0
10 R G16     2B36EEC45420BE12
R G17     0000000000009A8C
R G18     2471657C6826B413
R G19     FFFFFFFFFFEC14
R G20     0B878BECE9CB81EB
15 R G21     4B8BB7DB5A819822
R G22     1802F99900C52857
R G23     0000000000009AE8
R G24     33945300C89F9CA8
R G25     BB6468800821D2BA
20 R G27     0000000000002EAC
R G28     EF15111260248D64
R G30     0000000000004F30
R G31     B4281C2F814ECB1D
R HID0    0000000100000000
25 R HID1    F91C000000000000
R HID4     8B60000000000011
R HID5     8000000000000010
R IAR      EF15111260248D8C
R LR       000ADB9349BD5A65
30 R MSR      D000000000002800
R SRR0     EF15111260248D64
R SRR1     D000000000002800
R MMCR0    0000000080000000
R PIR      00000000
35 R RESRV    D85A4A6C5226E350
R SDAR     BF7CE4C07B15444A
R SIAR     CCBBA3970BFEEB78
R SDR1     0000000000040000
R_SIM      0000000000000000
40 END_OF_PROCESS
PROCESS 1
INITIALIZATIONS: REGISTERS
R ACCR     0000000000000000
R CacheConfig 0000000100800080
45 R CR       E80C16DA
R CTRL     00000001
R CTR      03050DDADAE158F3
R DABR     2EDFBBF4F145D2AE
R DAR      3199CE3A5F630C33
50 R DSISR    DF1E77CB
R XER      00005E7DF050001E
R FPSCR     80053012
R HID0     0000000100000000
R HID1     F91C000000000000

```

```

R HID4      8B6000000000001
R HID5      800000000000010
R IAR       B9C941307EAF2AC
R LR        00080C71A70B574F
5  R MSR      D000000000002000
   R MMCR0    0000000080000000
   R PIR      00000001
   R RESRV    BFDBA561D425EB30
   R SDAR     E29A1C5FF01E316B
10  R SIAR    A1CC7362F0AA681C
   R SDR1     000000000040000
   R_SIM      000000000000000

PHASE 0  INSTRUCTIONS

15  EPILOGUE
   * Begin macro Epilogue_Sequence
   I 000001307EAF2AC 7C0004AC * EA=B9C941307EAF2AC sync_NE
   I 000001307EAF2B0 7C0004AC * EA=B9C941307EAF2B0 sync_NE
20  I 000001307EAF2B4 7C0004AC * EA=B9C941307EAF2B4 sync_NE
   I 000001307EAF2B8 F000006E * EA=B9C941307EAF2B8 notrace
   * End of macro Epilogue_Sequence
   *****

RESULTS: REGISTERS
25  R ACCR      0000000000000000
   R CacheConfig 0000000100800080
   R CR         E80C16DA
   R CTRL       00000001
   R CTR        03050DDADAE158F3
30  R DABR      2EDFBBF4F145D2AE
   R DAR        3199CE3A5F630C33
   R DSISR      DF1E77CB
   R XER        00005E7DF050001E
   R FPSCR      80053012
35  R HID0      0000000100000000
   R HID1      F91C000000000000
   R HID4      8B6000000000001
   R HID5      800000000000010
   R IAR       B9C941307EAF2BC
40  R LR        00080C71A70B574F
   R MSR      D000000000002000
   R MMCR0    0000000080000000
   R PIR      00000001
   R RESRV    BFDBA561D425EB30
45  R SDAR     E29A1C5FF01E316B
   R SIAR    A1CC7362F0AA681C
   R SDR1     000000000040000
   R_SIM      000000000000000
END_OF_PROCESS
50  END_OF_CLUSTER
RESULTS: DATA MEMORY (MEMORY)
D 0000000000003000 BB6468800821D2BA
D 0000000000003100 BE65C5F7DA43DFBE
D 0000000000003200 8EE3A86B37E2C5E9

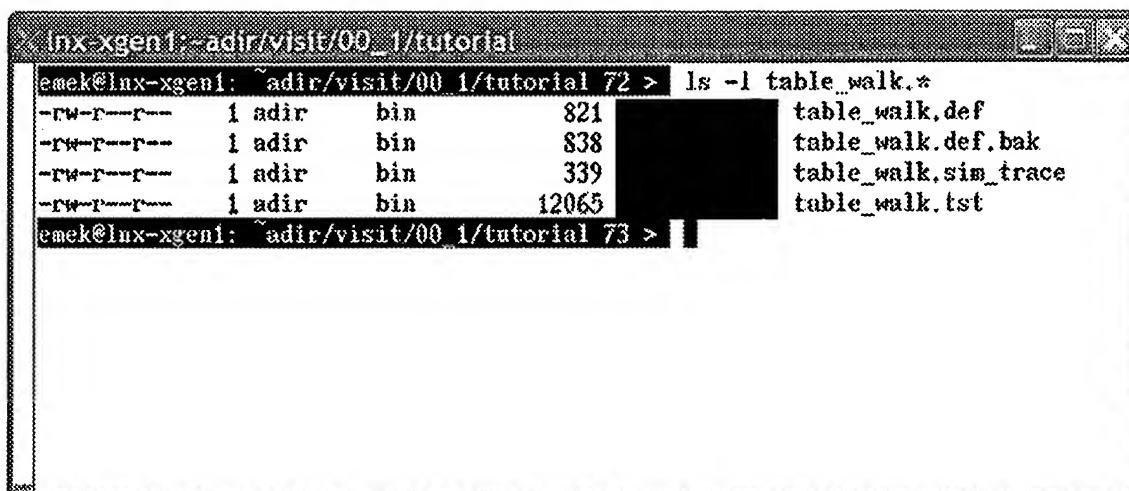
```

```

D 0000000000003300 471C748F37EBFD8A
D 0000000000003400 0B878BEC9CB81EB
D 0000000000003500 2471657C6826B413
D 0000000000003600 1802F99900C52857
5 D 0000000000003700 2B36EEC45420BE12
D 0000000000003800 5CC77810154A4BCF
D 0000000000003900 0BC1F2A5DF45CEF7
D 0000000000003A00 96CFED829B251CF8
D 0000000000003B00 9608082B9C2F3D9E
10 D 0000000000003C00 D3B5720D5D85E7D0
D 0000000000003D00 33945300C89F9CA8
D 0000000000003E00 DF16182FB73809E7
D 0000000000003F00 B4281C2F814ECB1D
D 0000000000004000 4B8BB7DB5A819822
15 TAG 000000000000300 1
TAG 000000000000310 0
TAG 0000000000003000 1
TAG 0000000000003100 0
TAG 0000000000003200 1
20 TAG 0000000000003300 0
TAG 0000000000003400 0
TAG 0000000000003500 0
TAG 0000000000003600 1
TAG 0000000000003700 0
25 TAG 0000000000003800 0
TAG 0000000000003900 0
TAG 0000000000003A00 1
TAG 0000000000003B00 1
TAG 0000000000003C00 1
30 TAG 0000000000003D00 1
TAG 0000000000003E00 1
TAG 0000000000003F00 1
TAG 0000000000004000 1
TAG 0000011260248D30 1
35 TAG 0000011260248D40 0
TAG 0000011260248D50 1
TAG 0000011260248D60 0
TAG 0000011260248D70 1
TAG 0000011260248D80 0
40 TAG 000001307EAF2A0 1
TAG 000001307EAF2B0 1
END_OF_TEST.

```

## Exhibit 4



A terminal window titled "lnx-xgen1: ~adir/visit/00\_1/tutorial" displays the output of the command "ls -l table\_walk.\*". The output lists four files with their permissions, owner, group, size, and name. The file names are partially obscured by a black redaction box.

Permissions	Owner	Group	Size	File Name	
-rw-r--r--	1	adir	bin	821	table_walk.def
-rw-r--r--	1	adir	bin	838	table_walk.def.bak
-rw-r--r--	1	adir	bin	339	table_walk.sim_trace
-rw-r--r--	1	adir	bin	12065	table_walk.tst

The terminal prompt shows the user "emek@lnx-xgen1" in the directory "~adir/visit/00\_1/tutorial" at prompt 72. The command "ls -l table\_walk.\*" is entered, and the output is displayed. The prompt then changes to 73.



E5-1

**Exhibit 5**

5

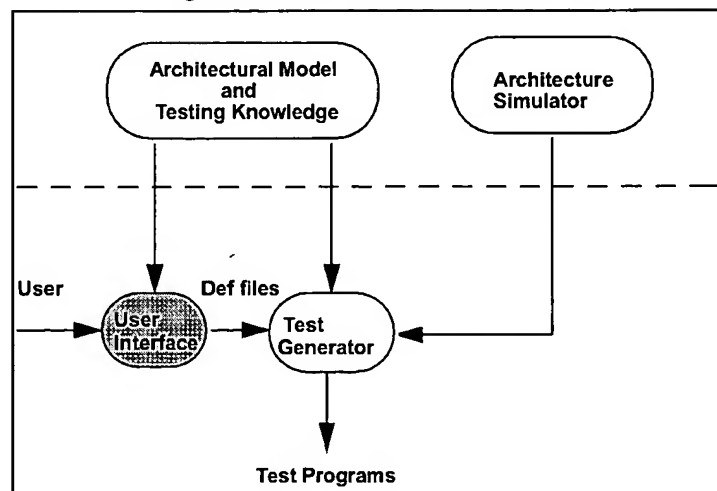
This page intentionally left blank.

## 1.1 Components of Genesys-Pro

The Genesys-Pro system consists of the following four basic interacting components, as illustrated in Figure 1:

- A generic, architecture-oblivious test generator engine.
- An external specification (the *model*) that holds a formal and declarative description of the targeted architecture. This specification is stored in a database which can also incorporate testing knowledge and some basic parameters of the microarchitecture. The integration of these components is commonly referred to as the Genesys-Pro *knowledge base*.
- A behavioral simulator that is used to predict instruction execution results, according to the architecture specification.
- A graphical user interface for defining the set of biasing directives to the test generators.

Figure 1. Main Components of Genesys-Pro



#### 4.6.2 Events

Events, similarly to macros, define a composite stream entity whose generation and execution are deferred to a later time. However, unlike macros, events are triggered by the state of the system satisfying some condition associated with the event, rather than by an explicit macro call.


Every event has an identifying name, a triggering condition, an optional reserved range start and size, and a stream entity that is to be generated. After every instruction is executed in the system, Genesys-Pro iterates over its list of events to see if their triggering condition is now TRUE. For each such event, Genesys-Pro generates and executes the associated stream entity. Note that the check for events is done by Genesys-Pro after the reentrancy check (See “Reentrancy” on page 55.).

Since generation of an event stream is not done until the triggering condition is met, users may want to pre-reserve the address space intended for the event (if there is such), to preclude inadvertent writes to these locations. They can do so by specifying the reserved range start and the reserved range size (See “Attributes” on page 42.). This will cause the desired address range to be reserved from the start of the test, regardless of whether the event is generated or not. This is particularly useful for interrupt handlers that reside at a particular fixed address in memory. For relocatable events, the reserved range start and size may be omitted.

Include an event in a def file in one of 2 ways:

- Directly in the def file.
- In a separate file and then include it in the def file.

**Include an event directly in a def file as follows:**

1. Select the **GPRODEF** statement (at the top of the tree) in the def file.
2. Click the **EVENT** statement button  in the Statements toolbar and insert **EVENT** as a child statement to the **GPRODEF** statement.
3. Double-click the **EVENT** statement in the def file and enter the name of the Event file.
4. Insert the statements to be included in the event, and specify its attributes.

**Define an event in a separate file and include it in the def file as follows:**

1. On the File menu, select **New**. The New Document dialog appears.
2. Select **Event file** and click **OK**. An untitled Event file is displayed.
3. Insert the statements to be included in the event, and specify its attributes. Name the event.
4. Save and name the event file.
5. To add the event file to the def file, double-click the **macroIncludeFile** statement while being on the **GPRODEF** statement in the def file, and enter the name of the Event file to be included.

Event conditions are simply expressions entered in the attribute window of the event. Similarly, **reservedRangeStart** (binary number) and **reservedRangeSize** (integer) are attributes of the event.

### 5.3 Interrupt Handler

When an interrupt occurs, Genesys-Pro continues with regular generation of instructions at the memory location defined by the interrupt. Since the same interrupt can occur many times, it is common to define an *interrupt handler* at the interrupt location. In real environments, such handlers are responsible for taking care of the cause of the interrupt and returning control to the main program afterwards (either at the same instruction or at the next one). In the Genesys-Pro environment, the primary purpose of handlers is to allow the generator to resume regular test generation. To achieve this, such handlers only need to return control to the main program to the next free location in which generation can resume.

Interrupt handlers are modeled as events in Genesys-Pro. As such, they can be:

- generated or verbatim
- fixed addressed or relocatable

- reentrant or single-shot - Reentrant is the natural property, but the user can limit the presence of a given interrupt to a single occurrence by *not* defining its handler as reentrant.

#### To create an interrupt handler:

1. Create an event (See "Events" on page 45.)
  2. Set the event condition attribute of the event:  
For fixed addressed interrupts, the condition is typically something like:  
"Exception() || Val(REGISTER, "PC", 0) == BITS("0x0000000000000300")".  
For relocatable interrupts, the PC test is omitted.
  3. If the interrupt handler is not relocatable, reserve its space by setting the reserved range start and size attributes of the event.
  4. Define the event body as a substream of the event. This stream can be any legal subprocess stream (including verbatim). Use a **Return From Interrupt** instruction to return to the main instruction stream.
  5. Set the reentrant attribute of the substream to TRUE if it is reentrant.
- Below is an example of an interrupt handler.

Figure 13. Example of Interrupt Handler

